A Matlab Program for Testing Quasi-Monte Carlo Constructions

by

Lynne Serré

A research paper presented to the
University of Waterloo
in partial fulfillment of the requirements for the degree of
Master of Mathematics
in
Computational Mathematics

Supervisor: Christiane Lemieux

Waterloo, Ontario, Canada, 2010

© Lynne Serré 2010

I hereby declare that I am the sole author of this research paper. This is a true copy of the research paper, including any required final revisions, as accepted by my readers.

I understand that my research paper may be made electronically available to the public.

Abstract

The Monte Carlo method uses a random sample of size n to approximate the value of an s-dimensional integral and has a convergence rate of $O(n^{-1/2})$. By using low-discrepancy sampling, the quasi-Monte Carlo (QMC) method achieves a convergence rate of $O(n^{-1}(\log n)^s)$, which is asymptotically faster than that of the Monte Carlo method. However, due to the dependence of the QMC error bound on the dimension s, this result doesn't hold at practical values of n for even relatively small values of s. Still, many empirical studies have shown that QMC can outperform Monte Carlo integration in high-dimensional problems at practical values of n. Empirical studies are often conducted by researchers to investigate the behaviour of QMC methods since the QMC error bound suffers from several practical limitations. This paper presents a Matlab program that implements some of the more commonly used numerical tests and provides a simple graphical summary of the results.

Contents

1 Introduction **1**

2 Background Material **3**

2.1 Low-Discrepancy Point Sets 3

2.2 QMC Error Bound 4

2.3 Randomized QMC 5

2.4 Effective Dimension 6

3 User's Guide **10**

3.1 Performance Measures 10

3.2 Numerical Experiments 11

3.2.1 Test Function g_1 12

3.2.2 Test Function g_2 12

3.2.3 Test Function g_3 13

3.2.4 Option Pricing 13

3.2.5 Mortgage-Backed Securities 15

3.3 Input Parameters for *runTests* 16

4 Experiments **19**

4.1 Generalized Faure Sequences 19

4.2 A First Experiment 21

4.2.1 Tests and Results 21

4.3 A Second Experiment 27

4.3.1 Tests and Results 27

5 Conclusion **36**

References **37**

Chapter 1

Introduction

Monte Carlo has become a widely used numerical method for approximating integrals, especially those in high dimensions, with no known closed-form solution. Given a real-valued function f defined on the s-dimensional unit hypercube $[0,1)^s$, the Monte Carlo estimator for the integral

$$I(f) = \int_{[0,1)^s} f(\mathbf{u})\, d\mathbf{u},$$

where $\mathbf{u} = (u_1, \ldots, u_s)$, is given by

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} f(\mathbf{u}_i),$$

where $\{\mathbf{u}_1, \ldots, \mathbf{u}_n\}$ is an independent and identically distributed random sample from the uniform distribution over $[0,1)^s$. Since $\mathbf{u}_i \sim U([0,1)^s)$, it is easy to see that $E[f(\mathbf{u}_i)] = I(f)$, and it follows that $\hat{\mu}$ is an unbiased estimator for $I(f)$. Applying the Strong Law of Large Numbers, the Monte Carlo estimator $\hat{\mu}$ converges almost surely to $I(f)$ as the number of function evaluations n tends to infinity. Additionally, by the Central Limit Theorem, the Monte Carlo integration error

$$I(f) - \hat{\mu} \approx N(0, \sigma/\sqrt{n})$$

for sufficiently large n, where

$$\sigma^2 = \int_{[0,1)^s} (f(\mathbf{u}) - I(f))^2\, d\mathbf{u}$$

is the variance of f. Thus, the Monte Carlo integration error is in $O(1/\sqrt{n})$.

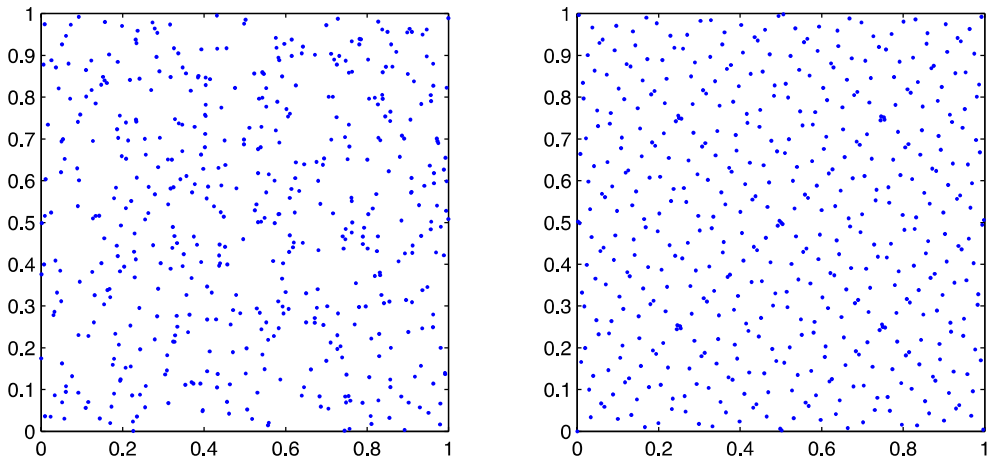


Figure 1.1: Both point sets contain 512 points, the one on the left is a random point set while the one on the right is a low-discrepancy point set (constructed using the Sobol' method).

The $O(1/\sqrt{n})$ convergence rate of the Monte Carlo error has the attractive property of being independent of the problem's dimension s, but also the unattractive property of being slow. One approach to speed up the performance of the Monte Carlo method is to replace the random sample used in a Monte Carlo estimator by a well-chosen deterministic point set, known as a *low-discrepancy* or *quasi-random* point set. This method is known as quasi-Monte Carlo integration. These low-discrepancy point sets are constructed to be more uniformly distributed than a random sample. In other words, low-discrepancy point sets are constructed to avoid the gaps and clusters that naturally occur in a random sample, as illustrated in Figure 1.1. The Koksma-Hlawka inequality establishes a deterministic bound on the quasi-Monte Carlo integration error, from which it follows that the quasi-Monte Carlo error convergence rate is in $O((\log n)^s/n)$. Thus, as long as n is sufficiently large, quasi-Monte Carlo converges faster than Monte Carlo. However, as we will see, the Koksma-Hlawka bound is subject to several limitations preventing its use as a practical error bound. As a result, researchers often turn to numerical experiments to test and compare quasi-Monte Carlo methods.

In this paper we present a Matlab program that facilitates the testing and comparison of quasi-Monte Carlo methods by means of numerical experiments. This program is detailed in Chapter 3. Chapter 2 presents the necessary background material on quasi-Monte Carlo integration, such as formally introducing low-discrepancy point sets and establishing the practical limitations of the quasi-Monte Carlo error bound. Lastly, in Chapter 4, we use our Matlab program to conduct some tests with a specific class of quasi-Monte Carlo constructions known as generalized Faure sequences.

Chapter 2

Background Material

2.1 Low-Discrepancy Point Sets

Quasi-Monte Carlo methods are based on deterministic point sets that are constructed to be more uniformly distributed than a random sample. The uniformity of a point set is measured by the concept of discrepancy. The star discrepancy is one of the more popular uniformity measures used in quasi-Monte Carlo methods, which we define following [6]. This discrepancy measure considers all hyper-rectangles in $[0,1)^s$ with one corner at the origin. More precisely, these hyper-rectangles are sets of the form

$$B(\mathbf{v}) = \{\mathbf{u} \in [0,1)^s : 0 \leq u_j \leq v_j, 1 \leq j \leq s\},$$

where $\mathbf{v} = (v_1, \ldots, v_s) \in [0,1)^s$. For a point set $P_n = \{\mathbf{u}_1, \ldots, \mathbf{u}_n\}$, let $\alpha(P_n, \mathbf{v})$ denote the number of points $\mathbf{u}_i \in P_n$ that are contained in $B(\mathbf{v})$. The empirical distribution F_n induced by P_n is then $F_n(\mathbf{v}) = \alpha(P_n, \mathbf{v})/n$, which is compared to the uniform distribution over $[0,1)^s$ via the Kolmogorov-Smirnov statistic, and yields the star discrepancy

$$D^*(P_n) = \sup_{\mathbf{v} \in [0,1)^s} |v_1 v_2 \cdots v_s - \alpha(P_n, \mathbf{v})/n|.$$

As we will see in the next section, the star discrepancy plays a central role in bounding the quasi-Monte Carlo integration error and so it will make sense to choose deterministic point sets with a low star discrepancy. The following lower bounds on the star discrepancy were given in [10], and have been cited by several other authors. It is conjectured for dimensions $s \geq 3$, but known for $s = 1$ and $s = 2$, that any point set P_n satisfies

$$D^*(P_n) \geq B_s n^{-1} (\log n)^{s-1},$$

and that the first n points of any sequence satisfy

$$D^*(P_n) \geq B'_s n^{-1} (\log n)^s,$$

where B_s and B'_s are positive constants depending only on s. There are several examples of sequences known to satisfy $D^*(P_n) \in O(n^{-1}(\log n)^s)$, some of the more popular names being Sobol', Halton, Niederreiter, and Faure, the last of which is presented in detail in Section 4.1. For this reason, a sequence $\mathbf{u}_1, \mathbf{u}_2, \ldots$ is called a low-discrepancy sequence if $D^*(P_n) \in O(n^{-1}(\log n)^s)$ and the point set P_n obtained from the first n points of such sequences is called a low-discrepancy point set.

2.2 QMC Error Bound

Recall that our goal is to approximate the integral of f by an average of function evaluations:

$$\int_{[0,1)^s} f(\mathbf{u})\, d\mathbf{u} \approx \frac{1}{n} \sum_{i=1}^{n} f(\mathbf{u}_i).$$

The Koksma-Hlawka inequality establishes a bound on the integration error:

$$\left| \frac{1}{n} \sum_{i=1}^{n} f(\mathbf{u}_i) - \int_{[0,1]^s} f(\mathbf{u})\, d\mathbf{u} \right| \leq V(f) D^*(\mathbf{u}_1, \ldots, \mathbf{u}_n), \tag{2.1}$$

where $\mathbf{u}_1, \ldots, \mathbf{u}_n$ are any points in $[0,1)^s$, provided that $V(f)$ is finite. $V(f)$ denotes the variation of f on $[0,1]^s$ in the sense of Hardy and Krause, which, following [10], is given by

$$V(f) = \sum_{k=1}^{s} \sum_{1 \leq i_1 < i_2 < \cdots < i_k \leq s} V^{(k)}(f; i_1, \ldots, i_k),$$

where $V^{(k)}(f; i_1, \ldots, i_k)$ denotes the application of

$$V^{(s)}(f) = \int_0^1 \cdots \int_0^1 \left| \frac{\partial^s f}{\partial u_1 \cdots \partial u_s} \right| du_1 \cdots du_s$$

to the restriction of f to the k-dimensional face $\{(u_1, \ldots, u_s) \in [0,1]^s : u_j = 1 \text{ for } j \neq i_1, \ldots, i_k\}$. Note however that it is not necessary for f to be differentiable for $V^{(s)}(f)$ to be finite [5], and we refer the reader to [10] for a more general definition of $V^{(s)}(f)$, as well as for details on the Koksma-Hlawka bound.

It follows from the Koksma-Hlawka inequality that the quasi-Monte Carlo integration error is in $O(n^{-1}(\log n)^s)$ since the points $\mathbf{u}_1, \ldots, \mathbf{u}_n$ are chosen to form a low-discrepancy

point set. The probabilistic error bound for Monte Carlo integration is in $O(n^{-1/2})$, from which we can conclude, for fixed dimension s, that Monte Carlo converges more slowly than quasi-Monte Carlo. However, for $n^{-1}(\log n)^s$ to be smaller than $n^{-1/2}$ requires n to be huge in even relatively small dimensions. For example, in 10 dimensions this inequality holds once $n \approx 10^{39}$ [6]. This limits the usefulness of the quasi-Monte Carlo error bound in practice since 10^{39} is an infeasible sample size.

Another practical limitation of the Koksma-Hlawka inequality is the condition that $V(f)$ must be finite. This condition can be violated by even rather simple functions, such as

$$f(u_1, u_2) = \begin{cases} 0 & \text{if } u_1 \leq u_2 \\ 1 & \text{otherwise,} \end{cases}$$

as well as by functions arising in practical applications [6]. For instance, for $V(f)$ to be finite requires that f be bounded, which is often not the case with integrands in financial applications like option pricing [5]. It is important to note that $V(f)$, as well as $D^*(P_n)$, are difficult to compute. Even if it is possible to compute $D^*(P_n)$ and $V(f)$ (and $V(f)$ is finite), the Koksma-Hlawka inequality only provides an upper bound and so does not give a reliable error estimate. For these reasons, the quasi-Monte Carlo error bound is a poor practical error bound. Thus, in practice, quasi-Monte Carlo methods are often tested and compared by running numerical experiments. Still, it is desirable to have a method for obtaining error estimates. Error estimates can be obtained by applying a randomization to the point set, a technique known as randomized quasi-Monte Carlo, which is introduced in the next section.

2.3 Randomized QMC

In Monte Carlo integration it is straightforward to obtain error estimates because the Monte Carlo estimator is based on an independent and identically distributed (i.i.d.) sample of points. Quasi-Monte Carlo integration replaces the random sampling used in Monte Carlo methods by low-discrepancy sampling in order to obtain a faster rate of convergence, but the Koksma-Hlawka bound (2.1) does not provide reliable error estimates. Randomized quasi-Monte Carlo applies a randomization to the point set P_n while trying to preserve its low discrepancy, and thus tries to combine the best features of the Monte Carlo and quasi-Monte Carlo methods. By randomizing P_n, we can create several i.i.d copies of the quasi-Monte Carlo estimator to obtain an error estimate.

Basing our discussion on [6], let $P_n = \{\mathbf{u}_1, \ldots, \mathbf{u}_n\} \subseteq [0,1)^s$ be a deterministic low-discrepancy point set and $\widetilde{P}_n = \{\tilde{\mathbf{u}}_1, \ldots, \tilde{\mathbf{u}}_n\} \subseteq [0,1)^s$ denote the randomized point set obtained by applying a randomization function to each point $\mathbf{u}_i \in P_n$. The randomization

function should be such that each point $\tilde{\mathbf{u}}_i \in \widetilde{P}_n$ is $U([0,1)^s)$ so that the estimator based on $\widetilde{P}_n$ is unbiased:

$$E\left[\frac{1}{n}\sum_{i=1}^{n} f(\tilde{\mathbf{u}}_i)\right] = \frac{1}{n}\sum_{i=1}^{n} E\left[f(\tilde{\mathbf{u}}_i)\right] = \frac{1}{n}\sum_{i=1}^{n}\int_{[0,1)^s} f(\tilde{\mathbf{u}}_i)\, d\tilde{\mathbf{u}}_i = I(f).$$

It is also important that the randomization function does not destroy the low discrepancy of P_n, since then the advantage of using quasi-Monte Carlo over Monte Carlo is lost. With an appropriately chosen randomization, we can create a random sample of quasi-Monte Carlo estimators

$$\hat{\mu}_{rqmc,l} = \frac{1}{n}\sum_{\tilde{\mathbf{u}}\in\widetilde{P}_{n,l}} f(\tilde{\mathbf{u}}),$$

where $\widetilde{\mathbf{P}}_{n,1}, \ldots, \widetilde{\mathbf{P}}_{n,m}$ are m independent randomized copies of P_n. We can then approximate $I(f)$ by the unbiased estimator

$$\hat{\mu}_{rqmc} = \frac{1}{m}\sum_{l=1}^{m} \hat{\mu}_{rqmc,l}$$

and estimate its variance by

$$\hat{\sigma}^2_{rqmc} = \frac{1}{m}\left(\frac{1}{m-1}\sum_{l=1}^{m}(\hat{\mu}_{rqmc,l} - \hat{\mu}_{rqmc})^2\right).$$

The variance estimate can then be compared with those obtained from other randomized quasi-Monte Carlo estimators or from the Monte Carlo estimator based on nm points. Randomization methods are generally designed for specific types of low-discrepancy point sets, and there are several common methods, but here we skip the details of specific methods and refer the interested reader to the literature, for instance, [5] or [6].

2.4 Effective Dimension

The quasi-Monte Carlo error convergence rate is $O(n^{-1}(\log n)^s)$, which is faster than the Monte Carlo convergence rate of $O(n^{-1/2})$. However, as mentioned, when s is large, for $n^{-1}(\log n)^s$ to be smaller than $n^{-1/2}$ requires an impractically large sample size n. For this reason, it was thought that quasi-Monte Carlo integration should not be used in high-dimensional problems. Some authors thought that quasi-Monte Carlo should not be applied in dimensions greater than 40, while others thought that it should not be applied in dimensions greater than 12 or 15 [5]. Despite what the theory suggested, Paskov and

Traub in [15] applied quasi-Monte Carlo integration to a high-dimensional finance problem and found quasi-Monte Carlo to be superior to Monte Carlo in dimensions as large as 360 at practical sample sizes. The concept of effective dimension was proposed to explain the success of quasi-Monte Carlo integration in high dimensions, but before we can define this concept, we must first define the ANOVA decomposition of a function.

As explained in [6], the functional ANOVA decomposition expresses an s-dimensional function f as a sum of 2^s components,

$$f(\mathbf{u}) = \sum_{I \subseteq \{1,\ldots,s\}} f_I(\mathbf{u}).$$

Each term in the sum, $f_I(\mathbf{u})$, is a function depending on a subset $(u_{i_1}, \ldots, u_{i_d})$ of the variables where $I = \{i_1, \ldots, i_d\} \subseteq \{1, \ldots, s\}$, and, for $I \neq \emptyset$, is given by

$$f_I(\mathbf{u}) = \int_{[0,1)^{s-d}} f(\mathbf{u})\, d\mathbf{u}_{-I} - \sum_{J \subset I} f_J(\mathbf{u}),$$

where $d = |I|$ and $-I = \{1, \ldots, s\} \setminus I$. Note that $f_\emptyset(\mathbf{u}) = I(f)$. Additionally,

$$\int_{[0,1)^s} f_I(\mathbf{u}) = 0$$

for all $I \neq \emptyset$ and, for all $I \neq J$, the ANOVA components are orthogonal:

$$\int_{[0,1)^s} f_I(\mathbf{u}) f_J(\mathbf{u})\, d\mathbf{u} = 0.$$

The ANOVA decomposition can be used to assess the difficulty of evaluating $I(f)$ by looking at the components' variance

$$\sigma_I^2 = \int_{[0,1)^s} f_I^2(\mathbf{u})\, d\mathbf{u}$$

and how their variance contributes to the overall variance of f:

$$Var(f) = Var(f(\mathbf{U})) = \sigma^2 = \sum_I \sigma_I^2.$$

The effective dimension provides a way of summarizing which of the components explain most of the function's variability. Here we state the definition of effective dimension as given in [6].

Definition 1 *The effective dimension of f in the superposition sense (and in proportion p) is the smallest integer d_S such that*

$$\frac{1}{\sigma^2} \sum_{I:|I|\leq d_S} \sigma_I^2 \geq p.$$

The effective dimension of f in the truncation sense (and in proportion p) is the smallest integer d_T such that

$$\frac{1}{\sigma^2} \sum_{I:i_d\leq d_T} \sigma_I^2 \geq p,$$

where $I = \{i_1, \ldots, i_d\}$.

In other words, a function with an effective dimension d_S can be well approximated by a sum of functions each depending on at most d_S variables while a function with an effective dimension d_T can be well approximated by a sum of functions each depending on only the first d_T variables $u_1, \ldots, u_{d_T}$ [6]. To illustrate the difference between the superposition and truncation senses, consider the function $f = \sum_{i=1}^{s} 2^i u_i$, which has $d_S = 1$ and $d_T = s$ [2]. The effective dimension depends on the proportion p, and in the truncation sense also depends on the order of the variables. Note that, for fixed p, $d_S \leq d_T$.

The low effective dimension, in one or both senses, of an integrand is often used to explain the success of quasi-Monte Carlo on high-dimensional problems. This was the case for the finance problem studied by Paskov and Traub; its effective dimension is much smaller than 360. If the projections over the first d_T coordinates of the chosen point set have good space-filling properties, then it is reasonable to expect that the point set will perform well on an integration problem with effective dimension in the truncation sense d_T [4]. If the projections over d_S-dimensional subspaces or less exhibit good space-filling properties, the point set should do well integrating a function with effective dimension in the superposition sense d_S [4]. Although low-discrepancy point sets are constructed to be more evenly distributed than a random point set, several low-discrepancy sequences are known to have poor space-filling properties in certain subspaces of $[0, 1)^s$. It is not uncommon for a sequence's low dimensional projections to be much more evenly distributed than those in higher dimensions, making them especially suitable for integrating functions of low effective dimension. For instance, as illustrated in Figure 2.1, the uniformity of the Halton sequence is known to deteriorate as the dimension increases [4]. The projection onto the first and second coordinates covers the unit square quite well, whereas the projection onto the 49th and 50th coordinates is concentrated on the unit diagonal. Since its projection over the first 2 coordinates is well distributed, the Halton sequence may perform well for a function with $d_T = 2$, but it may not perform so well for a function with $d_S = 2$ since its 2-dimensional projections deteriorate in higher dimensions. Depending on the properties

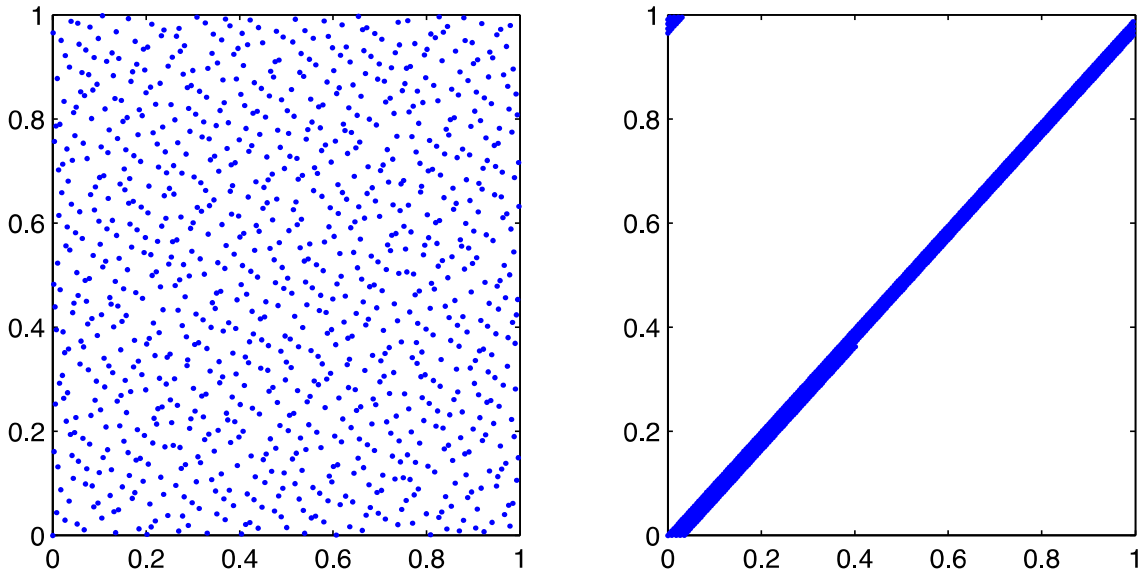


Figure 2.1: First 1000 points of the Halton sequence for the 1st and 2nd coordinates (left), and 49th and 50th coordinates (right).

of the chosen point set, some quasi-Monte Carlo methods will be more sensitive to the effective dimension of the integrand than others.

A closed-form expression for the ANOVA components f_I and their variance σ_I^2 is usually not available since, in particular, this requires knowing the value of $I(f)$ [6]. However, algorithms are available to approximate these quantities, two approaches are discussed in Section 6.3.3 of [6].

To summarize, the effective dimension can be used to help understand the difficulty level of the integration problem and to predict the performance of quasi-Monte Carlo methods. Quasi-Monte Carlo integration has been seen to outperform Monte Carlo integration for functions with a small effective dimension in the truncation sense, as well as for functions with a very small effective dimension in the superposition sense, but a large effective dimension in the truncation sense. Still, a low effective dimension is not sufficient to guarantee that quasi-Monte Carlo methods will be effective. Other factors, such as the dimension s, number of points n, and the choice of point set P_n, affect the performance of quasi-Monte Carlo integration. [17]

Chapter 3

User's Guide

As discussed in the previous chapter, the quasi-Monte Carlo error bound has several limitations preventing its use as a practical error bound. Consequently, in practice, numerical experiments are needed to test the performance of quasi-Monte Carlo constructions. In this chapter, we present a tool that facilitates the empirical testing and comparison of quasi-Monte Carlo constructions. Our tool is a Matlab function, called *runTests*, which is detailed in Section 3.3. First, we present the performance measures and numerical experiments available to the user.

3.1 Performance Measures

There are two main types of integrands that can be found in the literature for assessing the performance of quasi-Monte Carlo constructions [4]. The first type consists of test-functions g for which the solution to

$$I(g) = \int_{[0,1)^s} g(\mathbf{x})\, d\mathbf{x}$$

is known. In this case, the absolute error

$$E_n = |I(g) - \hat{\mu}|$$

can be computed exactly and analyzed as a function of the number of function evaluations n in the quasi-Monte Carlo estimator

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} g(\mathbf{x}_i),$$

where $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subseteq [0,1)^s$ is a low-discrepancy point set.

The second type consists of integrands g arising from practical problems, examples of such problems in finance are given in the next section. The exact solution to the integral is usually unknown for this type of problem and therefore must be estimated if the absolute error is to be used as a performance measure. An estimate of the exact value of $I(g)$ can be obtained by running a simulation based on a very large value of n. However, running such large simulations to obtain an error estimate may not be realistic for practitioners and so randomized quasi-Monte Carlo may be a more practical approach [4].

There are three measures available in the Matlab tool. The first is the absolute error

$$E_n = |I(g) - \hat{\mu}|,$$

which is a suitable measure for deterministic sequences. If m randomizations are applied to the sequence, then the average absolute error,

$$E_{n,avg} = \frac{1}{m} \sum_{l=1}^{m} |I(g) - \hat{\mu}_{rqmc,l}|,$$

or the estimated variance of the randomized sequences,

$$\hat{\sigma}_n^2 = \frac{1}{m-1} \sum_{l=1}^{m} (\hat{\mu}_{rqmc,l} - \hat{\mu}_{rqmc})^2,$$

can be used as performance measures where $\hat{\mu}_{rqmc,l}$ and $\hat{\mu}_{rqmc}$ are the randomized quasi-Monte Carlo estimators defined in Section 2.3.

3.2 Numerical Experiments

Our Matlab tool implements five numerical experiments. The first three, which we refer to as test-functions g_1, g_2, and g_3, can be computed exactly and have been used in several comparative studies on quasi-Monte Carlo constructions, see, for example, [4, 17, 9, 18, 16]. The last two test functions arise from practical problems in financial mathematics: the pricing of options and mortgage-backed securities. As mentioned in the previous chapter, it was originally thought that quasi-Monte Carlo methods were not suitable for high-dimensional problems, but empirical tests have shown otherwise. One such example is the mortgage-backed security problem, which can be represented as a 360-dimensional integral. Financial problems in the quasi-Monte Carlo setting have been studied in numerous papers, an extensive list is given in Section 5.5 of [5].

3.2.1 Test Function g_1

The first test function is

$$g_1(\mathbf{x}) = \prod_{j=1}^{s} \frac{|4x_j - 2| + a_j}{1 + a_j},$$

where $\mathbf{x} = (x_1, \ldots, x_s)$ and $I(g_1) = 1$. There are five parameter choices available to the user:

1. $a_j = 0.01$,
2. $a_j = 1$,
3. $a_j = j$,
4. $a_j = j^2$,
5. $a_j = (s - j + 1)^2$,

for $1 \leq j \leq s$. With parameter choices (1) and (2), all variables are equally important, whereas with parameter choices (3) and (4) the influence of x_j decreases as j increases [18]. Choice (5), when compared to (4), essentially reverses the order of the coordinates of $\mathbf{x}$, and so the influence of x_j increases as j increases. Changing the parameters from choice (1) to (4) decreases the effective dimension in the truncation sense of the integrand. Choice (5) has a large effective dimension in the truncation sense, probably equal to s. This function is useful for testing the sensitivity of low-discrepancy constructions to the effective dimension in the truncation sense of the integrand. [4]

3.2.2 Test Function g_2

The second test function is

$$g_2(\mathbf{x}) = \prod_{j=1}^{s} [1 + c(x_j - 0.5)],$$

where $\mathbf{x} = (x_1, \ldots, x_s)$ and $0 < c \leq 1$ is a constant to be chosen by the user. Changing the constant c, changes the effective dimension of the integrand. As noted in [4], it is more appropriate to measure the difficulty of this function by its effective dimension in the superposition sense, rather than the truncation sense, since all variables x_j contribute to g_2 in the same way. A detailed analysis of this integrand, for instance the relation between the product sc and the efficiency of quasi-Monte Carlo integration in different dimensions, can be found in [16]. Note that for any choice of s and c, $I(g_2) = 1$.

3.2.3 Test Function g_3

The third test function is

$$g_3(\mathbf{x}) = \alpha_s \pi^{-s/2} \cos\left(\sqrt{\frac{1}{2}\sum_{j=1}^{s}[\Phi^{-1}(x_j)]^2}\right),$$

where Φ^{-1} denotes the inverse of the cumulative distribution function of a standard normal random variable. The constant α_s is determined numerically (in Maple) so that $I(g_3) = 1$. This function is part of a family of isotropic integrals (see [12]). Papageorgiou and Traub show in [14] that quasi-Monte Carlo beats Monte Carlo for this type of integrand, and Owen suggests in [12] that this result is explained by the low effective dimension of the integrand.

3.2.4 Option Pricing

An option is a type of financial contract giving the owner the right, but not the obligation, to buy (call option), or sell (put option), one or more assets in the future according to specified terms and conditions. Here we consider the single asset case for two specific types of call options, Asian and digital. The problem at hand is to determine the price, or value at time 0, of the option. This price is given by the expected value (under a risk-neutral probability measure) of the option's discounted payoff

$$E[e^{-rT}P(S)],$$

where r is the risk-free interest rate, T is the expiration time of the option, and $P(S)$ is the option's payoff function, which depends on the price of the asset S [6].

The payoff function of a discretely monitored Asian option [6] is

$$P(S) = \max\left(0, \frac{1}{s}\sum_{j=1}^{s} S(t_j) - K\right),$$

where K, called the strike price, is the price at which the option holder can buy the asset at time T, and $S(t_j)$ is the price of the asset at time points $0 < t_1 < t_2 < \ldots < t_s = T$. For simplicity, we assume the time points are equally spaced, that is, $t_j = jT/s$ for $1 \le j \le s$. The number of time points at which the asset's price must be simulated determines the dimension s of the problem. Assuming $S(t_j)$ is lognormally distributed, the asset price at time t_j is generated under the risk-neutral measure as follows

$$\begin{aligned} S(t_j) &= S(t_{j-1})e^{(r-\sigma^2/2)T/s+\sigma\sqrt{T/s}\Phi^{-1}(u_j)} \\ &= S(0)e^{(r-\sigma^2/2)t_j+\sigma\sqrt{T/s}\sum_{i=1}^{j}\Phi^{-1}(u_i)}, \end{aligned} \tag{3.1}$$

where σ is the volatility of the asset and $\mathbf{u} = (u_1, \ldots, u_s) \sim U([0,1)^s)$. In the quasi-Monte Carlo setting, the point $\mathbf{u}$ is instead taken from a low-discrepancy point set. Thus, the goal is to compute the s-dimensional integral

$$E[e^{-rT}P(S)] = \int_{[0,1)^s} e^{-rT} \max\left(0, \frac{1}{s}\sum_{j=1}^{s} S(0)e^{(r-\sigma^2/2)t_j + \sigma\sqrt{T/s}\sum_{i=1}^{j}\Phi^{-1}(u_i)} - K\right) du_1 \ldots du_s,$$

which has no closed-form solution. In order to use the absolute error as a performance measure, an estimate for the above integral is obtained by using the digitally shifted Sobol' method with $n = 10^7$ points. The Sobol' method has been shown to be very effective in financial applications [5].

The payoff function of a digital option [13] is

$$P(S) = \frac{1}{s}\sum_{j=1}^{s}(S(t_j) - S(t_{j-1}))_+^0 S(t_j),$$

where

$$(S(t_j) - S(t_{j-1}))_+^0 = \begin{cases} 1 & \text{if } S(t_j) > S(t_{j-1}), \\ 0 & \text{otherwise.} \end{cases}$$

As with the Asian option, the asset prices $S(t_1), \ldots, S(t_s)$ are generated in chronological order using formula (3.1), which is the standard method, but these prices can be generated by other methods as well. One alternative is the Brownian bridge technique, which can also reduce the effective dimension of the integrand and improve the performance of quasi-Monte Carlo integration [6]. The author of [13] used the digital option to show that generating the asset prices with the Brownian bridge method does not always lead to a smaller error compared to the standard method.

In summary, to use the Asian option as a test function in our tool the user will need to provide the following quantities:

- the dimension s,
- the strike price K,
- the initial price, $S(0)$, of the asset,
- the risk-free rate r,
- the expiration time T of the option in years, and,
- the asset's volatility σ.

With the exception of the strike price, the same quantities are needed to use the digital option test function.

3.2.5 Mortgage-Backed Securities

A mortgage-backed security is a different type of financial contract, but, like an option, our goal is still to determine its present value. More specifically, "the goal is to evaluate the time-0 value of the cash flow received by the holder of a mortgage-backed security, which is a product that banks sell to investors and in which the cash flows come from the payments made by mortgage holders [6]." This problem has been used in several quasi-Monte Carlo studies. Here, we consider a simple case of the mortgage-backed security problem and follow the description and notation used in [2].

The time-0 value of the mortgage-backed security is given by

$$E\left[\sum_{k=1}^{M} u_k m_k\right],$$

where M is the length, in months, of the mortgage, u_k is the discount factor for month k, m_k is the cash flow for month k, and the expectation is taken with respect to the random variables involved in the interest rate fluctuations. The model used for the interest rate fluctuations is

$$i_k = K_0 e^{\xi_k} i_{k-1} = K_0^k e^{\xi_1 + \cdots + \xi_k} i_0,$$

where i_k denotes the interest rate for month k, $\xi_k \sim N(0, \sigma^2)$, $K_0 = e^{-\sigma^2/2}$ is a constant chosen so that $E[i_k] = i_0$, and i_0 is the interest rate at the beginning of the mortgage. The monthly discount factor and monthly cash flow are defined by

$$u_k = \prod_{j=0}^{k-1} (1 + i_j)^{-1} \quad \text{and} \quad m_k = c r_k[(1 - w_k) + w_k c_k],$$

respectively, where

$$\begin{aligned}
w_k &= \text{fraction of remaining mortgages prepaying in month } k, \\
r_k &= \prod_{j=1}^{k-1} (1 - w_j) \text{ is the fraction of remaining mortgages at month } k, \\
c &= \text{monthly payment, and} \\
c_k &= (\text{remaining annuity at month } k)/c = \sum_{j=0}^{M-k} (1 + i_0)^{-j}.
\end{aligned}$$

The prepayment rate w_k is modelled by

$$w_k = K_1 + K_2 \arctan(K_3 i_k + K_4)$$

where K_1, K_2, K_3, and K_4 are constants.

To summarize, the mortgage-backed security is specified by the parameters i_0, K_1, K_2, K_3, K_4, and σ^2. In our tool, there are three sets of parameters available:

$$\begin{aligned}(i_0, K_1, K_2, K_3, K_4, \sigma^2) &= (0.007, 0.01, -0.005, 10, 0.5, 0.0004),\\ (i_0, K_1, K_2, K_3, K_4, \sigma^2) &= (0.007, 0.04, 0.0222, -1500, 7.0, 0.0004),\\ (i_0, K_1, K_2, K_3, K_4, \sigma^2) &= (0.00625, 0.24, 0.134, -261.17, 12.72, 0.04),\end{aligned}$$

which will be referred to as "linear", "nonlinear", and "Tezuka", respectively. The first two sets of parameters were originally proposed in [2]. The "linear" set is so called because the resulting integrand is nearly a linear function of $u_1, \ldots, u_M$ where $\xi_k = \Phi^{-1}(u_k)$. The second set leads to an integrand which also has a large linear component, but not to the same extreme as the first, hence the name "nonlinear". The third set of parameters was originally proposed by Ninomiya and Tezuka in [11]. The dimension s of the integrand is equal to the length of the mortgage, M. In all cases, we consider 30-year mortgages, which create a 360-dimensional integral.

3.3 Input Parameters for *runTests*

In what follows, it is assumed that the reader is familiar with the basics of Matlab. The Matlab function *runTests* is used as follows:

runTests(inputFiles, testFnc, testFncType, N, interval, m, extraFiles).

The result of running this function is one or two graphs showing the values of one of the performance measures, described in Section 3.1, at various values of n for a series of QMC constructions and a particular test function. This function acts as a driver for several C programs that implement the test functions detailed in the previous section. The C programs use the RandQMC library, created by C. Lemieux, M. Cieslak, and K. Luttmer, which implements many QMC methods and their associated randomizations, as well as the Monte Carlo method. Details about this library, including a user's guide, are available at http://www.math.uwaterloo.ca/~clemieux/randqmc.html. Next we detail the information to be provided by the user.

inputFiles: a cell array of filenames corresponding to the input files needed by RandQMC. A detailed description of these files can be found at the website above; they specify, among other things, the QMC method to be used, the problem dimension, the randomization method, and the number of randomizations. At most 8 methods can be tested at once.

testFnc: a string specifying the test function to be used, which have been coded as 'g1', 'g2', 'g3', 'option', and 'mortgage'.

testFncType: some of the test functions require additional information, as follows.

- For 'g1', the test function type is an integer 1, 2, 3, 4, or 5 corresponding to a choice for the constants a_j described in Section 3.2.1.
- For 'g2', the test function type is the number $c \in (0, 1]$.
- For 'g3', the test function type is simply an empty string '' since no additional information needs to be specified. Note that this test function can only be used in dimensions 10, 20, 30, 40, 50, 120, 150, and 360.
- For 'option', the quantities listed at the end of Section 3.2.4 must be given in an array. For an Asian option, the test function type is the array $[s, K, S(0), r, T, \sigma]$. For a digital option, the array is $[s, S(0), r, T, \sigma]$.
- For 'mortgage', the test function type is either 'linear', 'nonlinear', or 'Tezuka' as described in Section 3.2.5. This function can only be used in dimension 360.

N: the number of points to be used in the QMC estimator (which has been denoted by n thus far).

interval: a number that specifies at which values of N the performance measure is to be plotted on the graph. For example, if N is 100,000 and the interval is set to 2000, then the value of the performance measure is plotted when the number of points used in the QMC estimator is 2000, 4000, 6000, ..., 98,000, and 100,000.

m: the number of randomizations. If $m = 1$, then a single graph showing the absolute error is produced. If $m > 1$, then two graphs are produced, one showing the average absolute error, a second showing the estimated variance.

extraFiles: some of the QMC methods in the library require an extra input file, their filenames are provided in a cell array.

To simplify things for the user, a GUI version of the function, *runTestsGUI*, is also available and prompts the user for all of the information described above. It has no input arguments and is run by typing *runTestsGUI()* at the Matlab command prompt. The GUI version acts a bit like a driver function for *runTests* in that instead of asking for the number of randomizations m, it asks the user for the number of graphs. So, it will either produce 1 graph (absolute error), 2 graphs (average absolute error and estimated variance), or 3 graphs (all measures). Note that different input files are needed based on which performance measure is to be used. The function *runTestsGUI* copies the input files but changes both m and the randomization method to 1 (no randomization) before

it creates a graph of the absolute error. This way, only one set of input files is needed when the user wishes to produce all 3 graphs at once. The function *runTests* gives the user control over the order of the QMC constructions in the plot, and may be more convenient to use when running many similar tests consecutively.

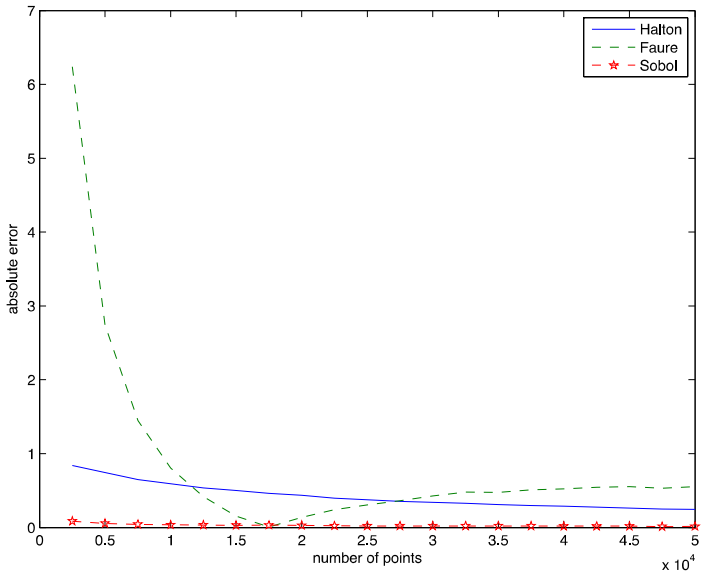


Figure 3.1: Absolute error for the mortgage-backed security problem (nonlinear case).

An example of the type of graph created by the Matlab function is given in Figure 3.1. To illustrate how the function is used, this graph was produced with the following input:

- inputFiles = {'inputHalton.dat', 'inputFaure.dat', 'inputSobol.dat'}
- testFnc = 'mortgage'
- testFncType = 'nonlinear'
- N = 50000
- interval = 2500
- m = 1
- extraFiles = {}

As a last detail, the cell array *inputFiles* is used to build the plot legend. So, the QMC methods are labelled by the name of their input file for the RandQMC library. If the user chooses to follow the naming convention input*SomethingDescriptive*.dat for their input files, then *SomethingDescriptive* will appear in the legend (see the above example). For the user more familiar with Matlab, short scripts can be written to customize the graphs, such as adding titles, changing the legend, or saving the graphs in various formats.

Chapter 4

Experiments

The Faure sequence is a well-known low-discrepancy sequence. The building block of this sequence (and others) is the base b expansion of the sequence's index i. In this chapter, we compare the performance of the Faure sequence and one of its generalizations, called the GF1 sequence, in different bases. We also compare the performance of the GF1 sequence with a newly proposed generalized Faure sequence.

4.1 Generalized Faure Sequences

Let $\mathbf{u}_i$ denote the ith point of a generalized Faure sequence in the prime base $b \geq s$ for $i \geq 0$. The ith point is obtained by first expressing i in its base b expansion

$$i = \sum_{l=0}^{\infty} a_l(i) b^l,$$

where $a_l(i) \in \mathbb{Z}_b$ and infinitely many $a_l(i)$ are zero. Then, for $1 \leq j \leq s$, the jth coordinate of $\mathbf{u}_i$ is given by

$$u_{ij} = \sum_{l=0}^{\infty} \tilde{a}_{j,l}(i) b^{-l-1},$$

where

$$(\tilde{a}_{j,0}(i), \tilde{a}_{j,1}(i), \ldots)^T = C_j (a_0(i), a_1(i), \ldots)^T$$

and $C_j = A_j P_b^{j-1}$ is a matrix of infinite dimensions with entries in $\mathbb{Z}_b$. A_j is a nonsingular lower triangular matrix and P_b is the upper triangular Pascal matrix defined by

$$P_b(r, c) = \binom{c}{r} \bmod b \quad \text{for } r, c = 0, 1, \ldots.$$

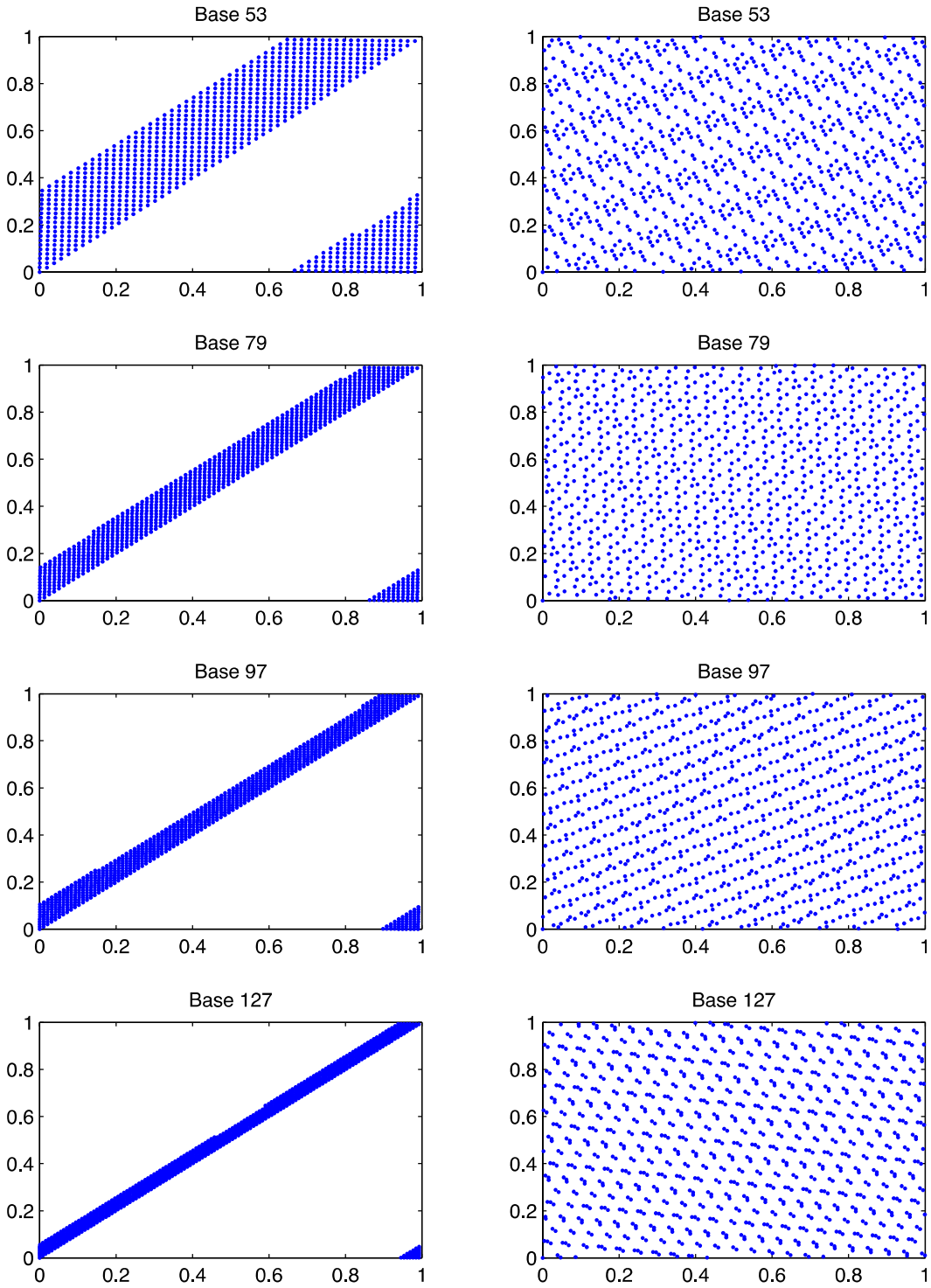


Figure 4.1: First 1000 points of the Faure (left column) and GF1 (right column) sequences in different bases for the 49th and 50th coordinates.

The original Faure sequence is obtained by taking A_j to be the identity matrix I for $1 \leq j \leq s$. When the number of points n is small and the dimension s is large, the space-filling properties of the Faure sequence may not be very good [6]. As explained in [9], "powers of the Pascal matrix have ones on their diagonal, which in turn implies that points from these sequences will tend to clump along the main diagonal in $[0,1)^s$ initially, and the larger b is, the more time it takes before this behaviour goes away." This behaviour of the Faure sequence is illustrated in the left-hand side of Figure 4.1. By multiplying the powers of the Pascal matrix by a matrix A_j, the generalized Faure sequence aims to improve the space-filling properties of the Faure sequence. We will consider two generalized Faure sequences in our experiments that both take $A_j = f_j I$ for $1 \leq j \leq s$ where the numbers f_j are carefully chosen factors. The first sequence, called the GF1 construction, was proposed by Lemieux and Faure in [9] and chooses the factors f_j by considering the quality of two-dimensional projections. Figure 4.1 shows that the first 1000 points of the GF1 sequence cover the unit square much more uniformly than those of the Faure sequence. The second sequence uses a new method proposed by C. Lemieux [7] to select the factors f_j. Some further details about these two sequences will be given in Section 4.3.

4.2 A First Experiment

Generalized Faure sequences belong to a more general class of low-discrepancy sequences called (t, s)-sequences, for which a bound on the discrepancy of the first n points is given by

$$D^*(P_n) \leq c_s(\log n)^s + O((\log n)^{s-1}) \text{ [6]}.$$

Since all low-discrepancy sequences have the same asymptotic upper bound for $D^*(P_n)$, these sequences are sometimes compared by studying the behaviour of the constant c_s, which may depend on the dimension s. For the Faure sequence, it is known that

$$c_s = \frac{1}{s!}\left(\frac{b-1}{\log b}\right)^s \text{ [4]}.$$

Since it is desirable for $D^*(P_n)$ to be as small as possible, usually the base b used to construct the Faure sequence is taken to be the smallest prime greater than or equal to s. However, since these are asymptotic results, it is not clear how the sequence behaves when s is large and n is relatively small. In this section, we use our tool to compare the performance of the Faure and GF1 sequences in different bases.

4.2.1 Tests and Results

As a first test, the function g_1 was used with dimension $s = 50$ and four different bases: 53, 79, 97 and 127. Figure 4.2 shows the absolute errors for case 1, $a_j = 0.01$. In the

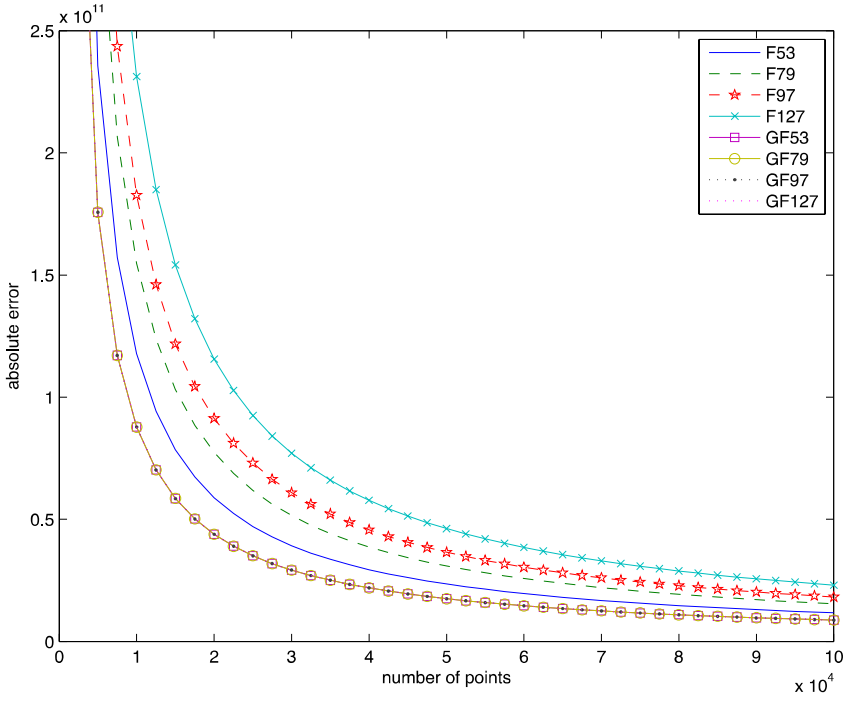


Figure 4.2: Absolute error for g_1 with $a_j = 0.01$ and $s = 50$.

figures, the Faure sequence is denoted by F, and the GF1 sequence by GF, followed by the base b. In this case, it is clear that for the Faure sequence, the larger the base, the larger the absolute error. For the GF1 sequence, there is so little difference in the absolute errors that the lines overlap for all bases. It seems that the Faure sequence is much more sensitive to the choice of base, which may agree with the behaviour of these sequences illustrated in Figure 4.1. For case 2, $a_j = 1$, the graph is very similar so it is not included here. For cases 3, 4, and 5, the variables are no longer equally important and the graphs reveal a different trend. These three graphs are also very similar, so only the results for case 3, $a_j = j$, are shown in Figure 4.3. In this case, for both sequences, the smallest base does not always give the smallest error. Note the difference in the scales for the absolute error between the two graphs in Figure 4.3.

The function g_1 was also used in dimension 75 with bases 79, 97, 127, and 367. For all five choices of a_j, the graphs showed trends similar to those in dimension 50. For the Faure sequence, base 367 stood out from the other bases as having an even larger absolute error for most values of n in all five cases.

For the g_2 function, two sets of parameters were tested, $(c, s) = (0.25, 96)$ and $(c, s) = (0.1, 120)$, which have been used in other studies, such as [4]. It is given in [4] that the

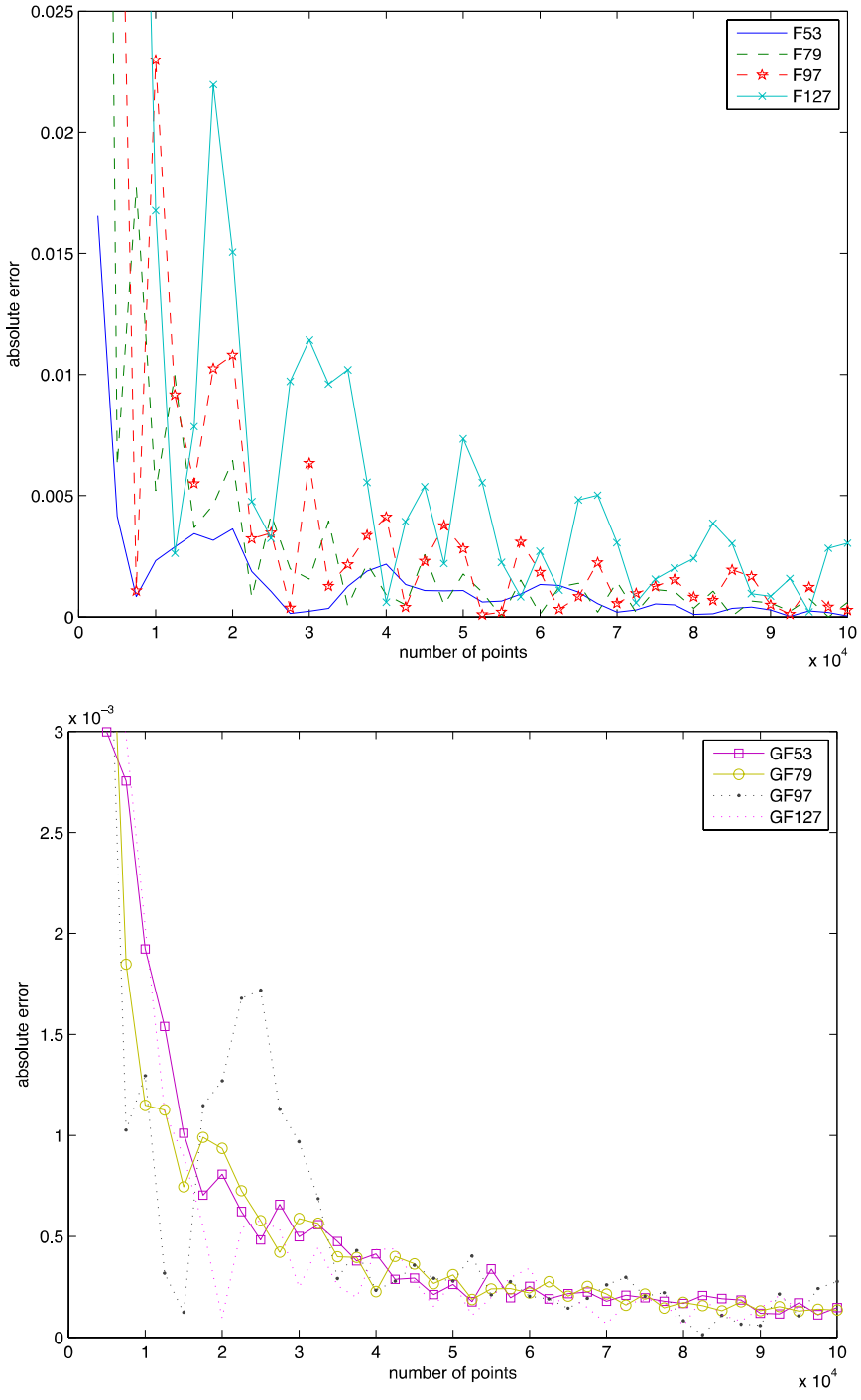
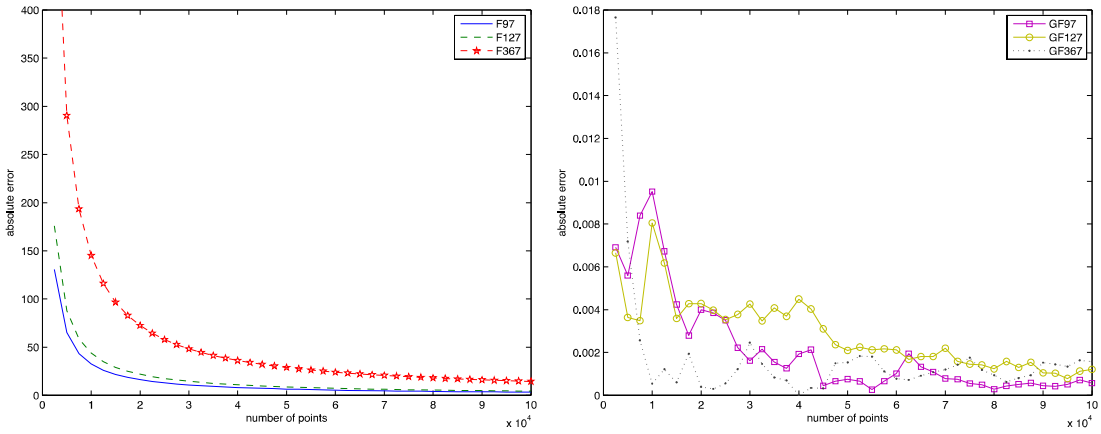


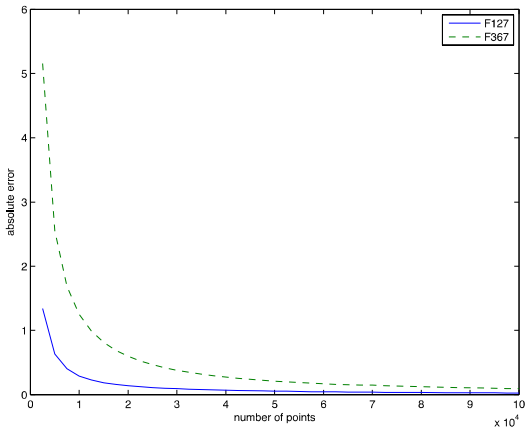
Figure 4.3: Absolute error for g_1 with $a_j = j$ and $s = 50$.

effective dimension in the superposition sense for each of these sets is 4 and 6, respectively. The results for the first set, $(0.25, 96)$, are shown in the top row of Figure 4.4, the second set in the bottom row of the same figure. For the Faure sequence, in both dimensions, since the lines never cross it is clear that a smaller base gives a smaller error. The situation is not so clear for the GF1 sequence, sometimes the largest base gives the smallest error. Note that the scale for the absolute error is quite different for each graph in Figure 4.4. In fact, the absolute errors for the GF1 sequence are always smaller than those for the Faure sequence. For $(c, s) = (0.25, 96)$, the absolute error of the Faure sequence is always greater than 3 (for base 367 it's always greater than 14), while the absolute error for the GF1 sequence is always smaller than 0.02.

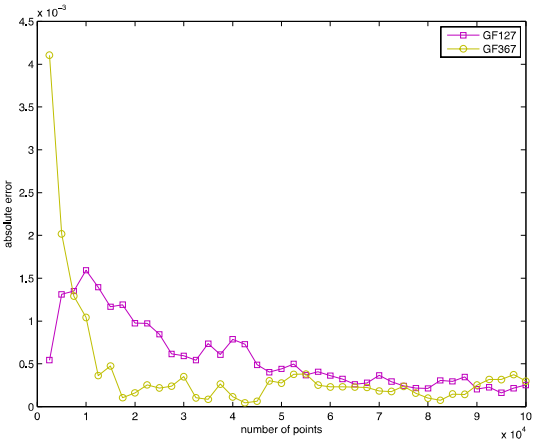


(a) Faure sequence, $c = 0.25, s = 96$

(b) GF1 sequence, $c = 0.25, s = 96$



(c) Faure sequence, $c = 0.1, s = 120$



(d) GF1 sequence, $c = 0.1, s = 120$

Figure 4.4: Absolute error for g_2.

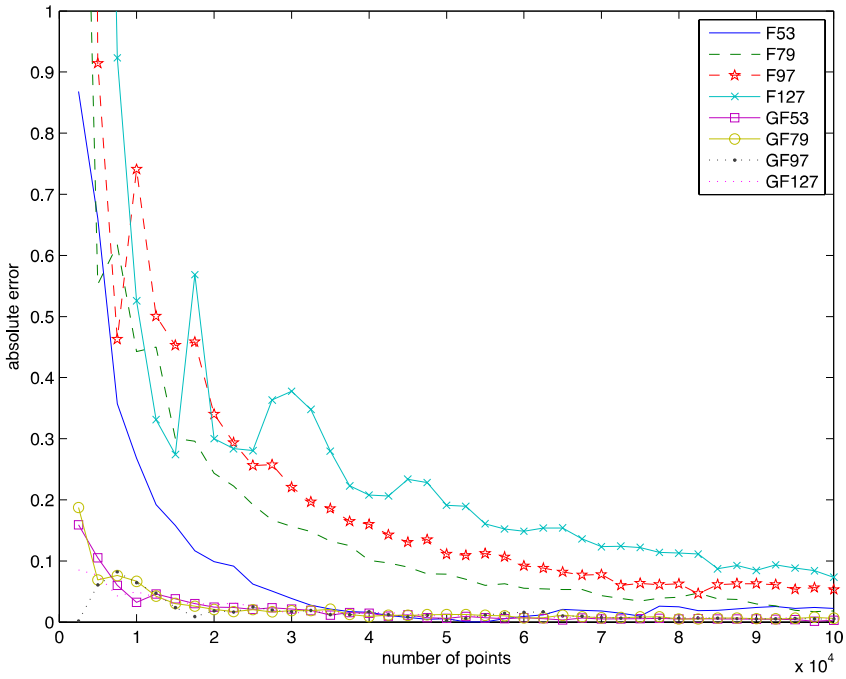
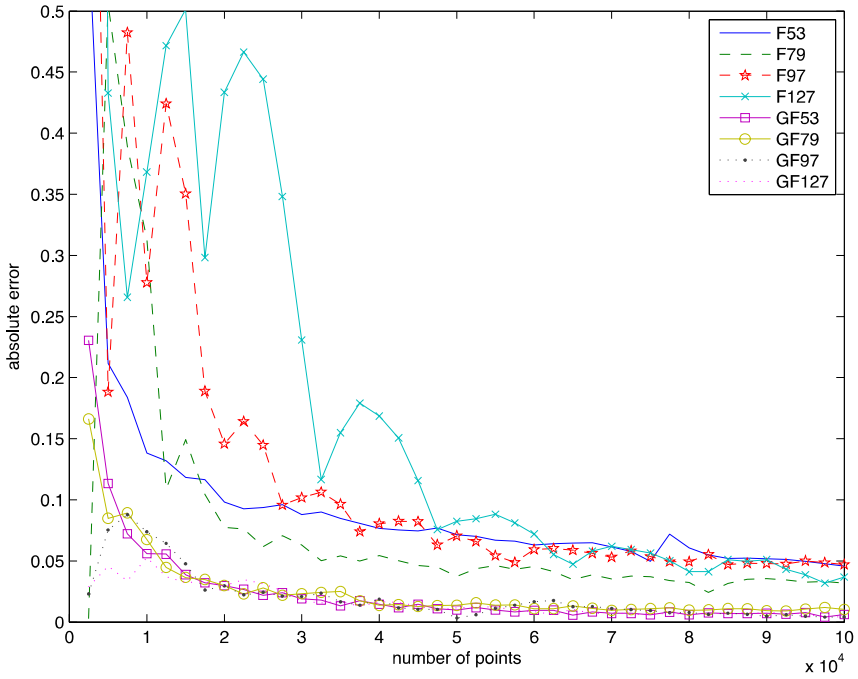


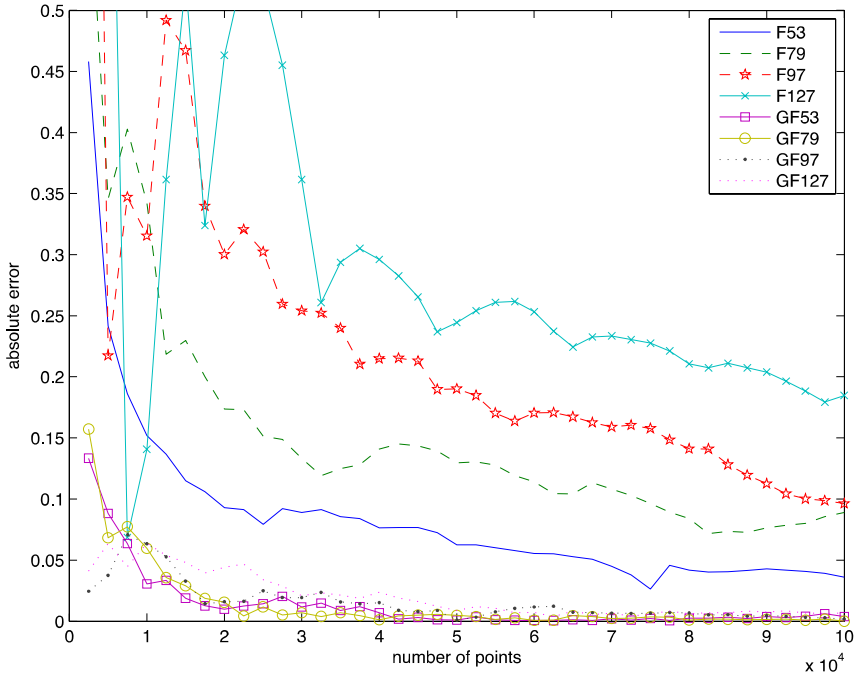
Figure 4.5: Digital option: $s = 50$, $S(0) = 100$, $r = 0.045$, $T = 1$, $\sigma = 0.3$.

The digital option was considered with $S(0) = 100$, $r = 0.045$, $T = 1$, and $\sigma = 0.3$, which are the same parameter choices used in [13]. The dimension was chosen to be $s = 50$ and the bases used were 53, 79, 97, and 127. The results are shown in Figure 4.5. Overall, there is much less variability in the accuracy of the GF1 sequence across different bases than in the accuracy of the Faure sequence. Once $n > 25,000$, the larger the base, the larger the absolute error of the Faure sequence. For this problem, there are several values of n for which the accuracy of the Faure sequence in base 53 is very similar to that of the GF1 sequence.

Lastly, the Asian option pricing problem in dimension $s = 50$ was used to compare the sequences' performance in different bases. In our experiment, $S(0) = 50$, $r = 0.05$, $T = 1$, $\sigma = 0.3$, and two different strike prices were considered, $K = 45$ and $K = 55$. The following four choices for the base were used: 53, 79, 97, and 127. The results are shown in Figure 4.6. For both choices of the strike price, there is a much larger difference between the accuracy of the Faure sequence in different bases than the GF1 sequence. For $K = 55$, once $n > 20,000$, the smaller the base used in the Faure sequence, the smaller the absolute error. For the smaller strike price, $K = 45$, once $n > 20,000$, the Faure sequence in base 79 actually does best.



(a) Strike price $K = 45$



(b) Strike price $K = 55$

Figure 4.6: Asian option: $s = 50$, $S(0) = 50$, $r = 0.05$, $T = 1$, $\sigma = 0.3$.

4.3 A Second Experiment

In this section, we compare the performance of two generalized Faure sequences. As mentioned, both sequences take $A_j = f_j I$ for $1 \leq j \leq s$, but differ in how the factors f_j are chosen. For the GF1 construction, as explained in [9], the first step in choosing the factors f_j is to establish a list containing the $b/2$ (approximately) best factors f based on the quantity θ_b^f. The criteria θ_b^f is a measure first proposed in [3] and provides a bound on the discrepancy of certain one-dimensional sequences. These factors give rise to good one-dimensional projections, but the idea behind this construction is to ensure that two-dimensional projections over nearby indices are also good. Thus, f_1 is chosen to be the best factor in the list (the one with the smallest θ_b^f) and, for $j = 2, \ldots, s$, the factor f_j is chosen from the list so that it minimizes

$$\tau_j(N_0, W) = \max_{1 \leq l \leq W} T(N_0, (S_b^{C_{j-l}}, S_b^C)),$$

where $C_{j-l} = f_{j-l} P_b^{j-l-1}$ and $C = f P_b^{j-1}$. The number W is the "window" size, it's the number of nearby indices being considered and, along with N_0, must be specified. $T(N_0, (S_b^{C_{j-l}}, S_b^C))$ is the L_2-discrepancy of the first N_0 points of the two-dimensional sequence formed by the $(j-l)$th and jth coordinates of the sequence under construction. The L_2-discrepancy is another type of discrepancy commonly used to measure the uniformity of a quasi-Monte Carlo construction. Unlike the star discrepancy, it is based on the L_2 norm instead of the L_∞ norm, which makes it much easier to compute. Also, it considers all hyper-rectangles in $[0, 1)^s$, not just those with a corner at the origin.

The second generalized Faure sequence we consider here uses a criteria that doesn't depend on a discrepancy measure to choose the factors f_j. This is a new method proposed by C. Lemieux [7]. In what follows, the factors used to construct this sequence will be referred to as the "new" factors, and the factors used to construct the GF1 sequence will be referred to as the "old" factors. Figure 4.7 shows the projections onto the 49th and 50th coordinates of these two generalized Faure sequences, with the sequence based on the new factors in the left column and the old factors in the right column. Both sequences improve upon the space-filling properties of the original Faure sequence (see Figure 4.1).

4.3.1 Tests and Results

Here we consider our generalized Faure sequences in four different bases: 53, 79, 97, and 127. The performances of the two sequences were compared in dimension $s = 50$ across all four choices of base. All five choices for the parameters a_j in the function g_1 were tested; the results for choices 3 and 5 are given in Figures 4.8 and 4.9, respectively. While the performance of the old factors (GF1) is quite similar across all four bases in dimension

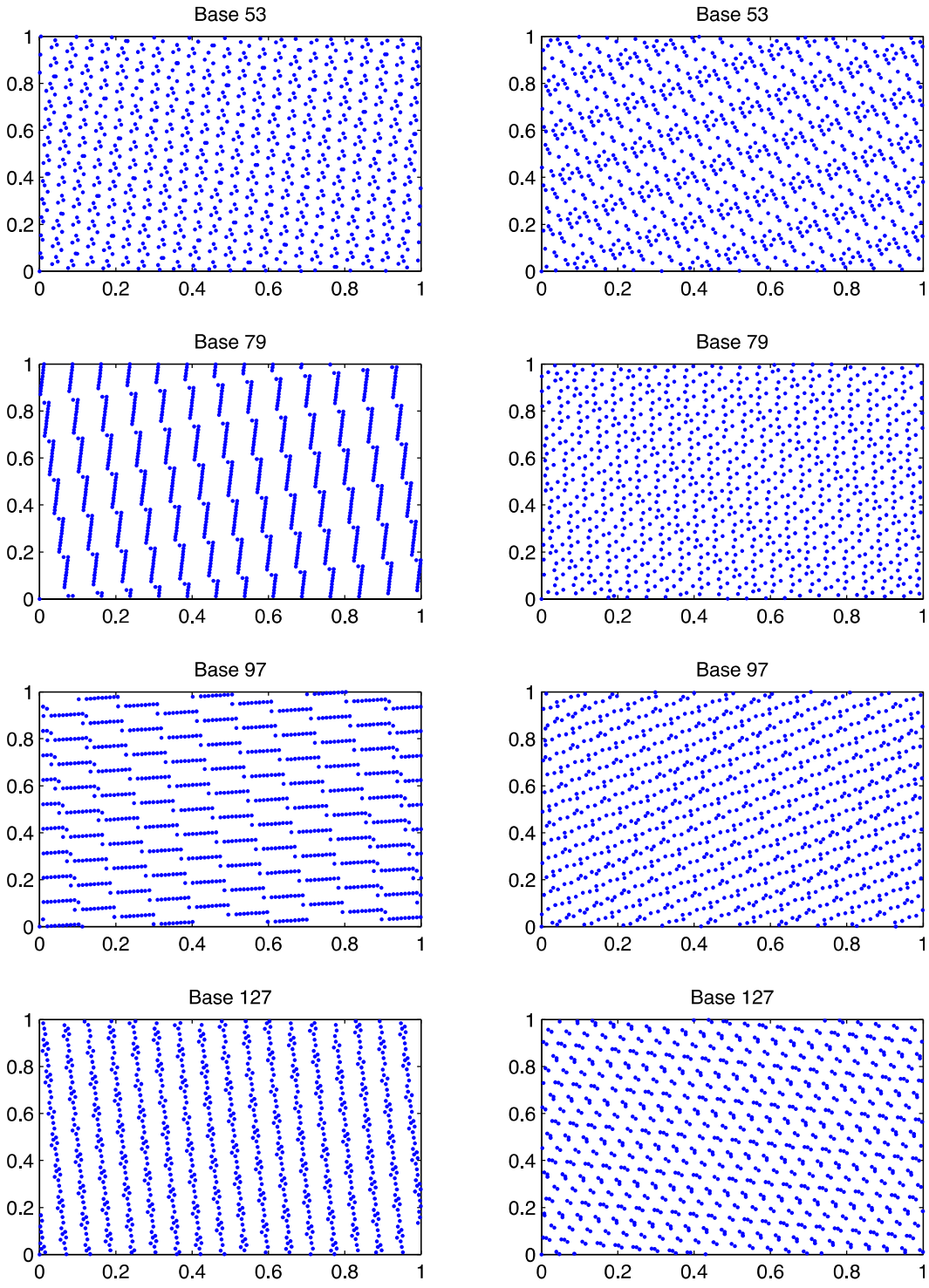


Figure 4.7: First 1000 points of the new generalized Faure (left column) and GF1 (right column) sequences in different bases for the 49th and 50th coordinates.

50, the performance of the new factors in dimension 50 seems to be more sensitive to the choice of base. This is particularly evident with parameter choice 5, $a_j = (50 - j + 1)^2$, where the absolute error for the new factors is considerably larger in bases 79, 97, and 127 than in base 53.

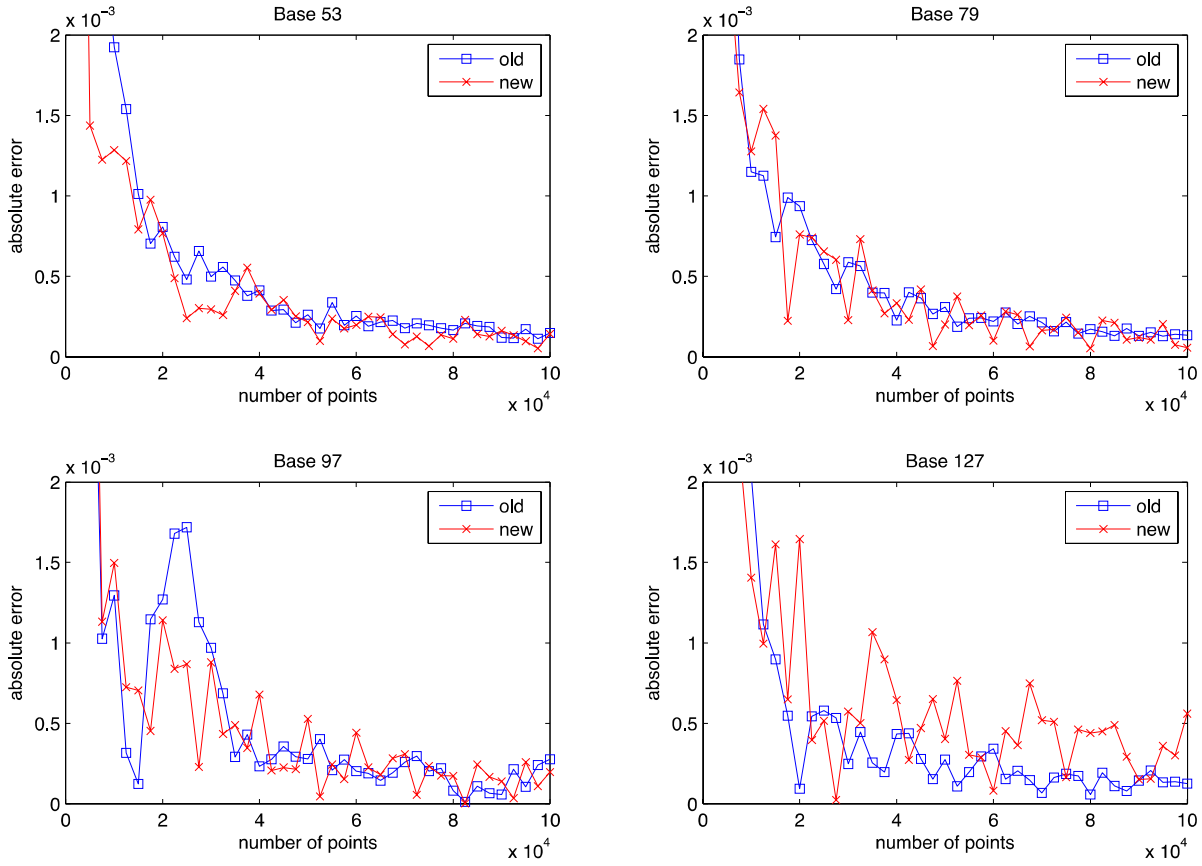


Figure 4.8: Absolute error for g_1 with $a_j = j$ (case 3) and dimension $s = 50$.

Figure 4.10 shows the absolute error for test function g_2 with $c = 0.25$, $s = 96$, and two choices for the base, 97 and 127. In base 97, the new factors usually have a slightly larger absolute error than the old factors, whereas in base 127 the new factors usually have a smaller absolute error. Figure 4.11 shows the absolute error for g_2 with $c = 0.1$, $s = 120$, and $b = 127$. In this case, the performance of both sequences is very similar.

The same practical integrands used in Section 4.2 were also used to compare the performance of the old and new factors. The results for the digital option are given in Figure 4.12 and the results for the Asian option, with $K = 45$ and $K = 55$, are given in Figure 4.13. The dimension s for both options was 50. For the digital option, the difference between

the absolute errors of the two sequences is fairly small, with the new factors having only slightly larger absolute errors. For the Asian option, the performance of both sequences is also very similar across all bases for both choices of the strike price.

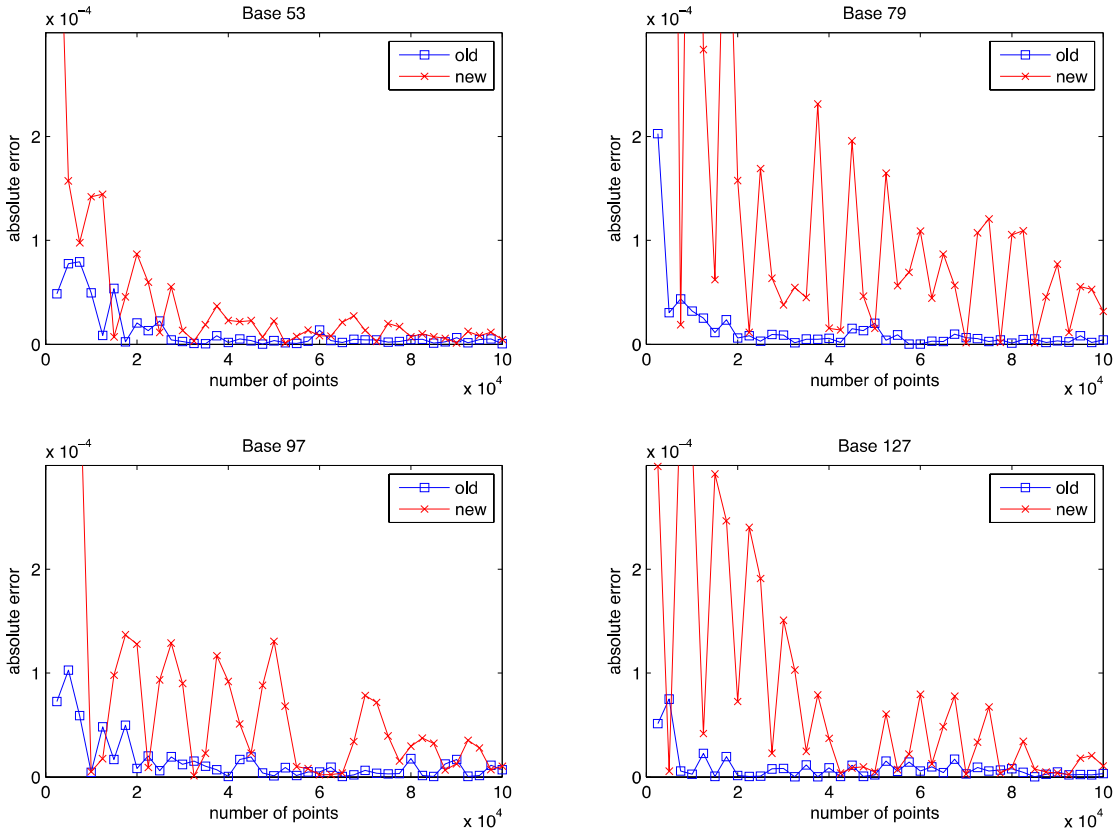


Figure 4.9: Absolute error for g_1 with $a_j = (s - j + 1)^2$ (case 5) and dimension $s = 50$.

Since some of the tests showed that the new factors do not perform as consistently across different bases as the old factors, these two sequences were also compared by selecting a dimension s and choosing the base b to be the smallest prime greater than or equal to s. The following four pairs (s, b) were used: (50, 53), (75, 79), (95, 97), and (125, 127). The results for test function g_1 with parameter choices 3 and 5 are shown in Figures 4.14 and 4.15, respectively. With parameter choice 3, the two sequences perform very similarly in all four dimensions. With parameter choice 5, the sequence based on the new factors generally has larger absolute errors than the sequence based on the old factors, the exception being $(s, b) = (125, 127)$ where the absolute errors are more similar.

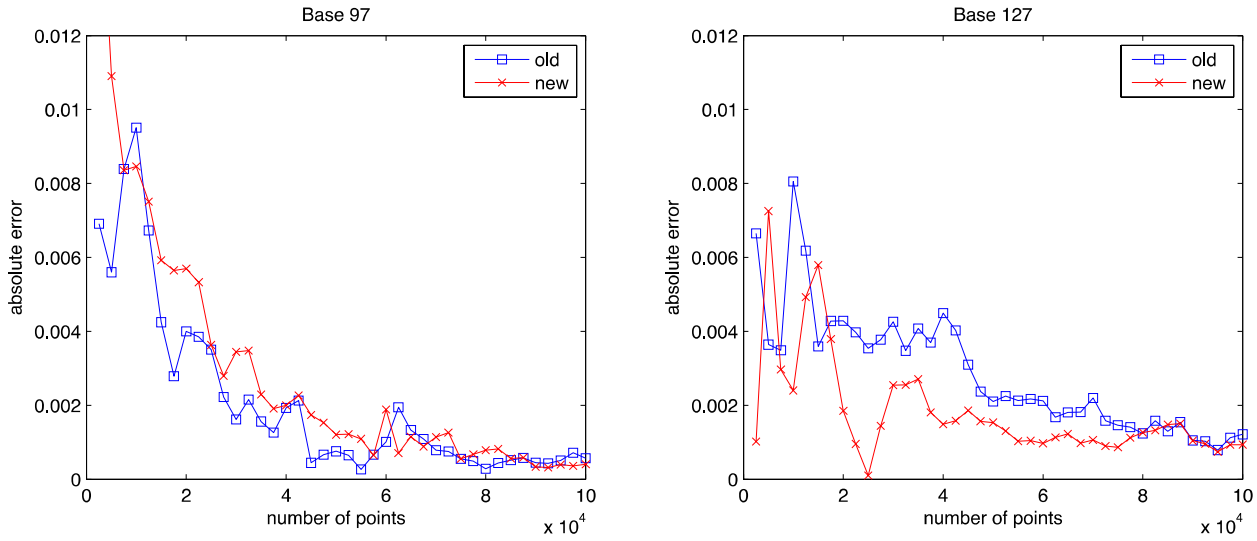


Figure 4.10: Absolute error for g_2 with $c = 0.25$ and dimension $s = 96$.

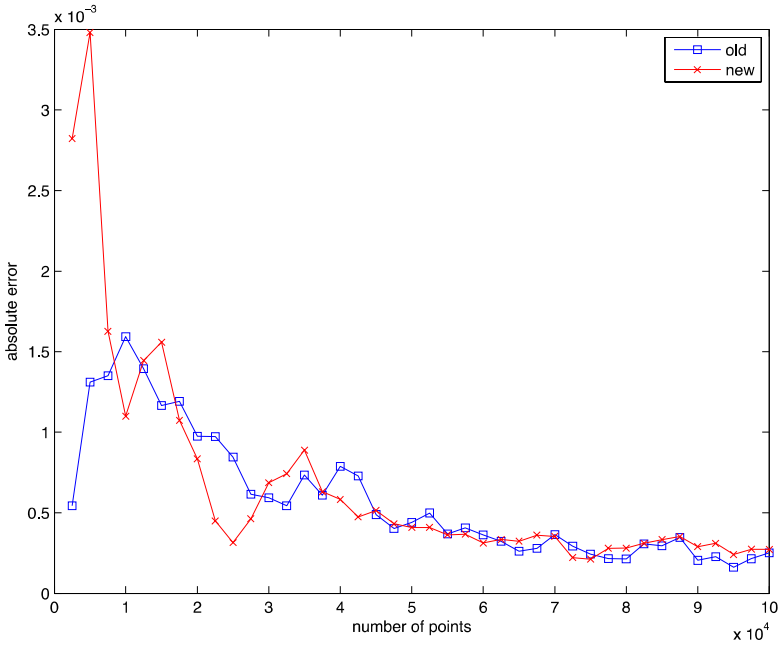


Figure 4.11: Absolute error for g_2 with $c = 0.1$, dimension 120, and base 127.

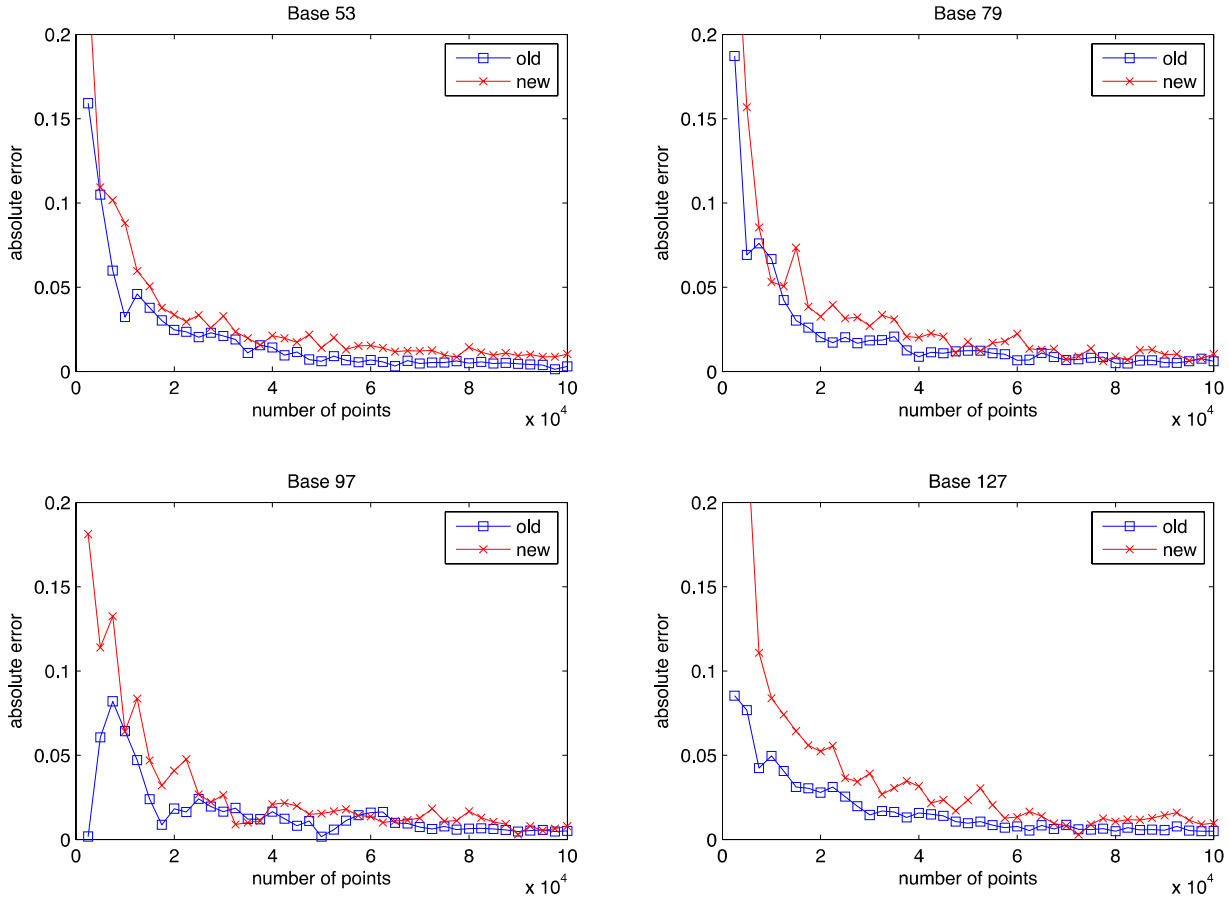
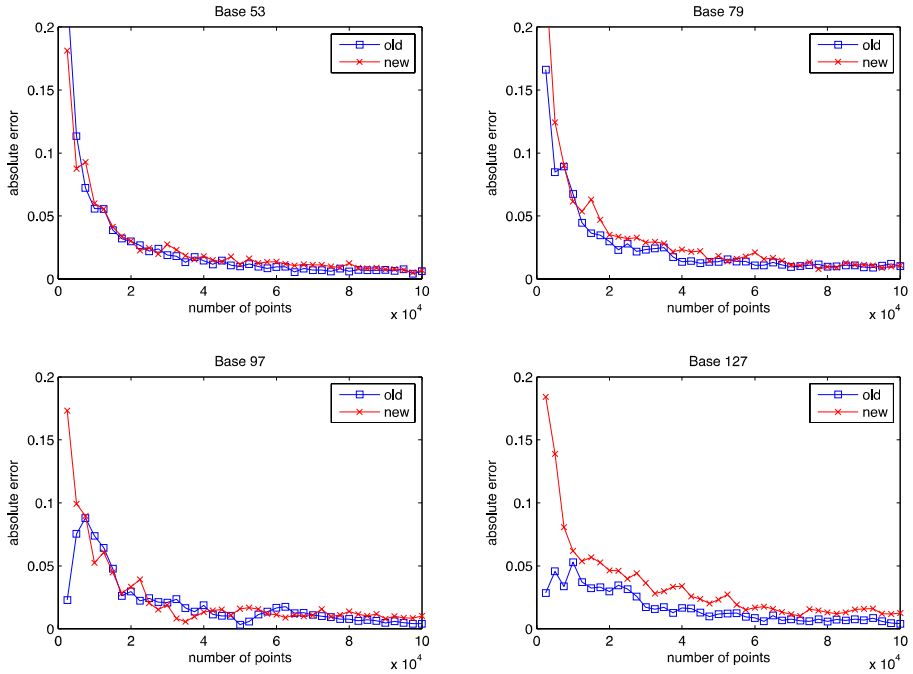
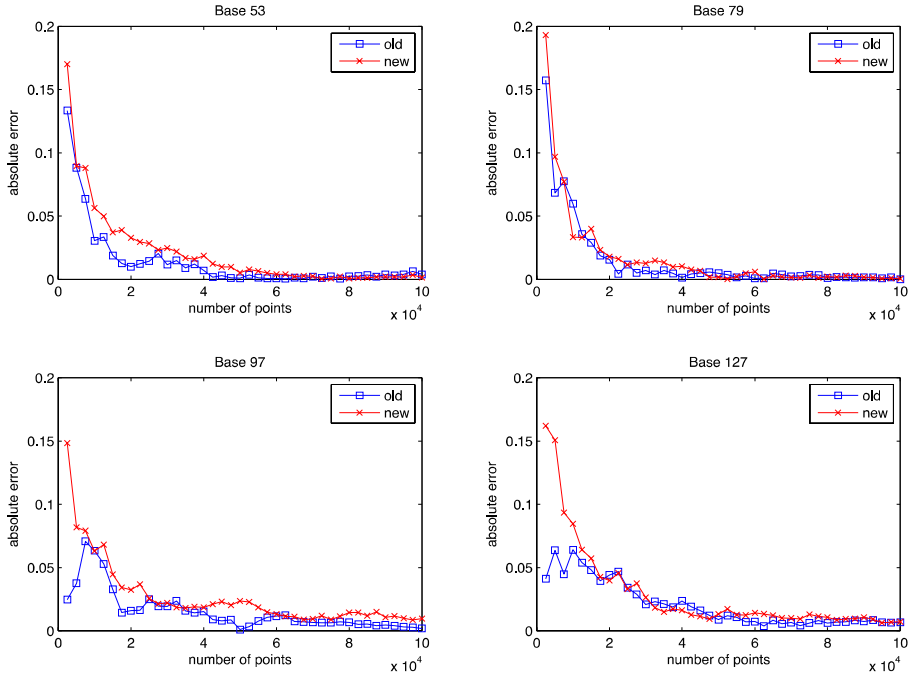


Figure 4.12: Digital option: $s = 50$, $S(0) = 100$, $r = 0.045$, $T = 1$, $\sigma = 0.3$.



(a) Strike price $K = 45$



(b) Strike price $K = 55$

Figure 4.13: Asian option: $s = 50$, $S(0) = 50$, $r = 0.05$, $T = 1$, $\sigma = 0.3$.

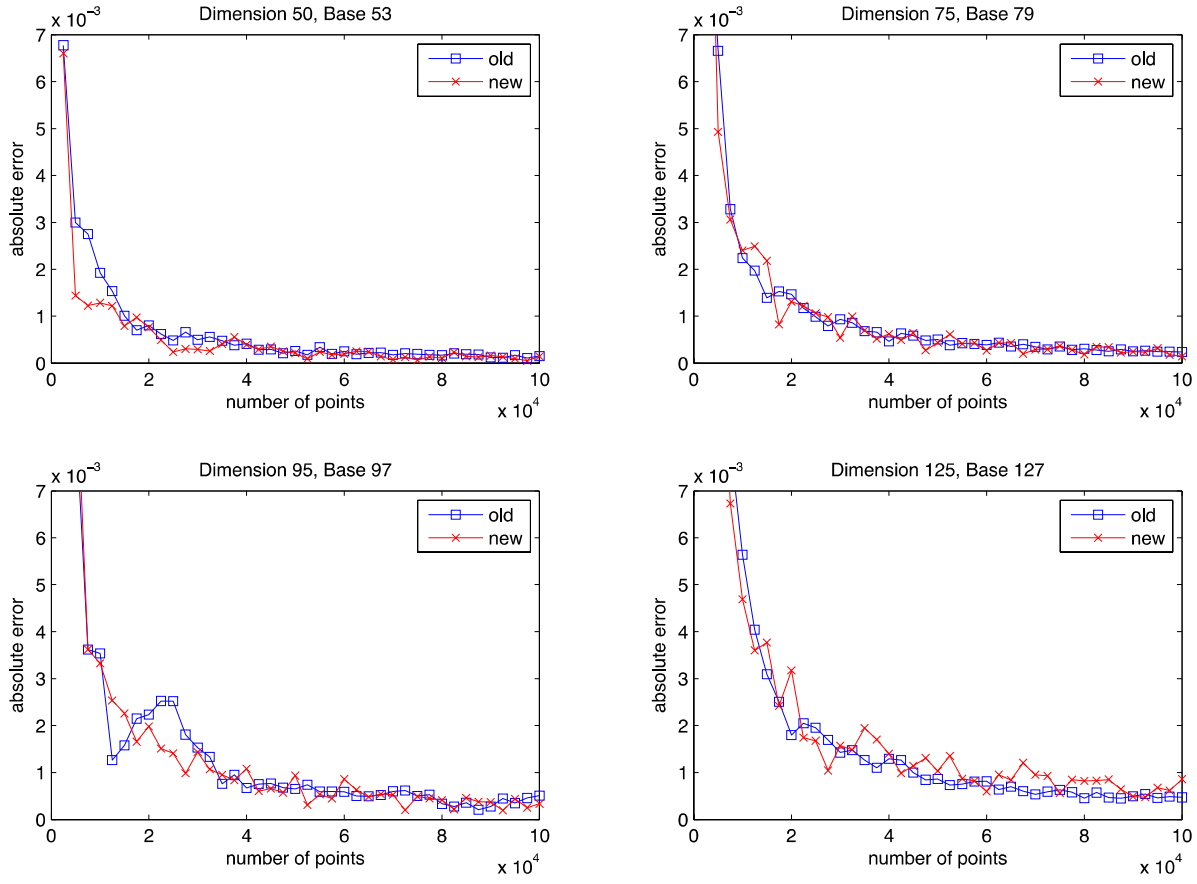


Figure 4.14: Absolute error for g_1 with $a_j = j$ (case 3).

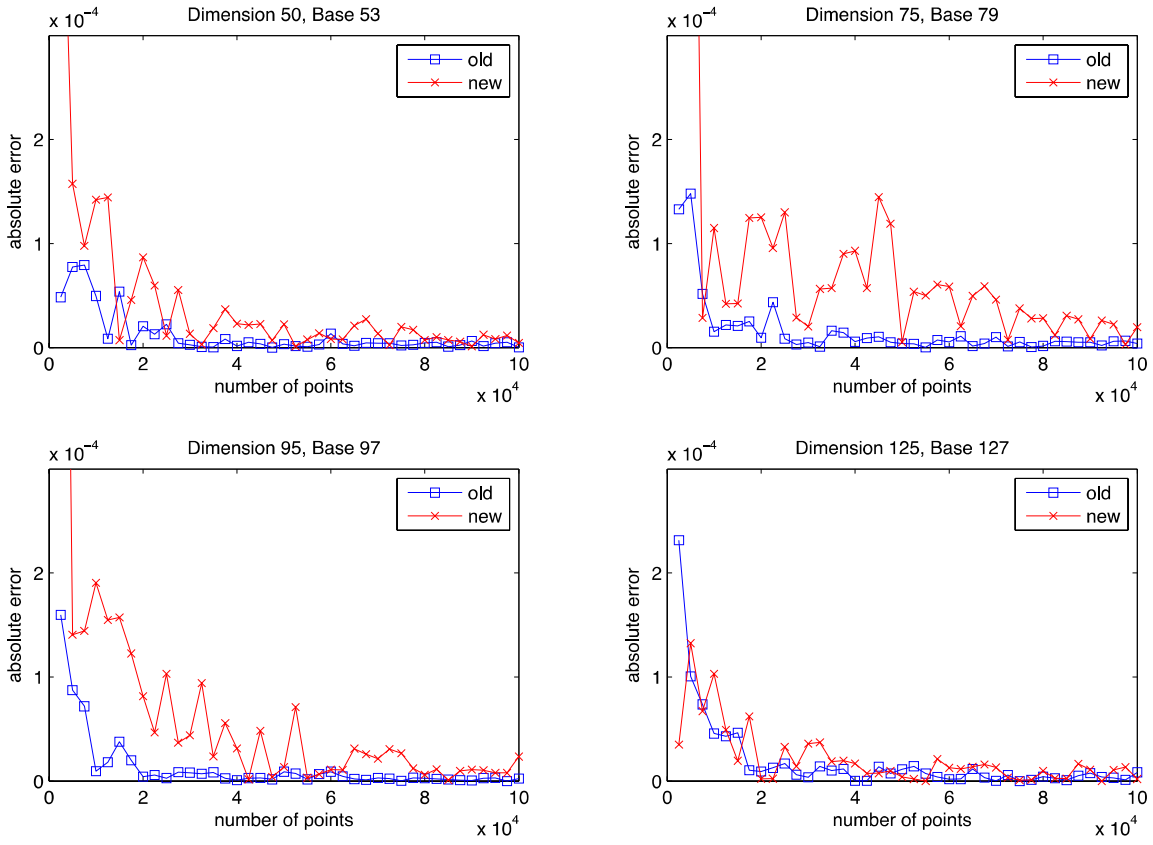


Figure 4.15: Absolute error for g_1 with $a_j = (s - j + 1)^2$ (case 5).

Chapter 5

Conclusion

Due to the limitations of the quasi-Monte Carlo error bound, as discussed in Section 2.2, researchers often use numerical experiments to test quasi-Monte Carlo constructions. In this paper, we presented a Matlab tool that facilitates the empirical testing and comparison of quasi-Monte Carlo constructions. The user has a choice of five numerical experiments, all of which have been used in several comparative studies on quasi-Monte Carlo methods. The first three experiments are "artificial" integrands for which an exact solution is known. By tuning their parameters, the user can test the sensitivity of a construction to different features of the integrand, such as its effective dimension. The remaining experiments are integrands arising from practical applications, such as pricing mortgage-backed securities, for which an exact solution is usually not available. A single Matlab function, *runTests*, gives the user access to all of the above-mentioned experiments, which have been implemented in C, and creates a graph allowing the simultaneous comparison of up to eight methods. Thus, our tool takes advantage of the computational power of C and the graphing capabilities of Matlab. The user also has the choice of testing constructions in a deterministic fashion (absolute error) or in a probabilistic fashion by applying an appropriate randomization method (average absolute error or estimated variance of the randomized sequence). A GUI implementation of *runTests* provides a user-friendly interface; it prompts the user for all the necessary input and could easily be used by someone with little working knowledge of Matlab.

Our Matlab tool was used to conduct some tests with two generalized Faure sequences based on carefully chosen factors f_j such that $A_j = f_j I$ for $1 \leq j \leq s$. Based on the results of the numerical tests, although the performances of the two sequences were similar in some cases, the newly proposed factors did not seem to provide any significant improvement over the factors used in the GF1 sequence. Thus, more work is needed to find improved quasi-Monte Carlo constructions.

References

[1] R. E. Caflisch. Monte Carlo and quasi-Monte Carlo methods. *Acta Numerica*, pages 1–49, 1998.

[2] R. E. Caflisch, W. Morokoff, and A. Owen. Valuation of mortgage-backed securities using Brownian bridges to reduce effective dimension. *Journal of Computational Finance*, 1(1):27–46, 1997. 8, 15, 16

[3] H. Faure. Selection criteria for (random) generation of digitial (0,s)-sequences. In H. Niederreiter and D. Talay, editors, *Monte Carlo and Quasi-Monte Carlo Methods 2004*, pages 113–126. Springer, 2006. 27

[4] H. Faure and C. Lemieux. Generalized Halton sequences in 2008: A comparative study. *ACM-TOMACS*, 19(4), 2009. 8, 10, 11, 12, 21, 22

[5] P. Glasserman. *Monte Carlo Methods in Financial Engineering*, volume 53 of *Application of Mathematics - Stochastic Modelling and Applied Probability*. Springer, 2004. 4, 5, 6, 11, 14

[6] C. Lemieux. *Monte Carlo and Quasi-Monte Carlo Sampling*. Springer Series in Statistics. Springer, 2009. 3, 5, 6, 7, 8, 9, 13, 14, 15, 21

[7] C. Lemieux. Private communication, July 2010. 21, 27

[8] C. Lemieux, M. Cieslak, and K. Luttmer. RandQMC user's guide: A package for randomized quasi-Monte Carlo methods in C. University of Calgary, 2004.

[9] C. Lemieux and H. Faure. New perspectives on (0,s)-sequences. In Pierre L'Ecuyer and Art B. Owen, editors, *Monte Carlo and Quasi-Monte Carlo Methods 2008*, pages 113–130. Springer, 2009. 11, 21, 27

[10] H. Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods*, volume 63 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. SIAM, 1992. 3, 4

[11] S. Ninomiya and S. Tezuka. Toward real-time pricing of complex financial derivatives. *Applied Mathematical Finance*, 3(1):1–20, 1996. 16

[12] A. B. Owen. The dimension distribution and quadrature test functions. *Statistica Sinica*, 13:1–17, 2003. 13

[13] A. Papageorgiou. The Brownian bridge does not offer a consistent advantage in quasi-Monte Carlo integration. *Journal of Complexity*, 18:171–186, 2002. 14, 25

[14] A. Papageorgiou and J.F. Traub. Faster evaluation of multidimensional integrals. *Computers in Physics*, 11:574–578, 1997. 13

[15] S.H. Paskov and J.F. Traub. Faster valuation of financial derivatives. *Journal of Portfolio Management*, 22(1):113–120, 1995. 7

[16] I.M. Sobol' and D.I. Asotsky. One more experiment on estimating high-dimensional integrals by quasi-Monte Carlo methods. *Mathematics and Computers in Simulation*, 62:255–263, 2003. 11, 12

[17] X. Wang and K-T Fang. The effective dimension and quasi-Monte Carlo integration. *Journal of Complexity*, 19:101–124, 2003. 9, 11

[18] X. Wang and F. J. Hickernell. Randomized Halton sequences. *Mathematical and Computer Modelling*, 32:887–899, 2000. 11, 12